Artículos originales

Sistema Multi-Agente para la Recomendación Personalizada de Tutores en U-Learning

Multi-Agent System for the Personalized Recommendation of Tutors in U-Learning

Luciano Gastón Juárez[1], Beatriz Fernández-Reuter[1], Elena Durán[1]

[1] Universidad Nacional de Santiago del Estero, Facultad de Ciencias Exactas y Tecnologías, Instituto de Investigaciones en Informática y Sistemas de Información, Santiago del Estero, Argentina.

lucianojuarezgaston@gmail.com, bfreuter@unse.edu.ar, eduran@unse.edu.ar

Recibido: 12/12/2020 | **Corregido:** 10/08/2021 | **Aceptado**: 04/01/2022

Cita sugerida: L. G. Juárez, B. Fernández-Reuter and E. Durán, "Sistema Multi-Agente para la Recomendación Personalizada de Tutores en U-Learning," *Revista Iberoamericana de Tecnología en Educación y Educación en Tecnología*, no. 32, pp. 18-27, 2022. doi: 10.24215/18509959.32.e2

Resumen

Los dispositivos como, teléfonos móviles y tablets son aliados de la educación, dado que permiten acceder a contenidos y actividades educativas desde cualquier lado y en cualquier momento. Sin embargo, el apoyo o asistencia al estudiante no siempre se encuentra disponible cuando un estudiante presenta un problema en su aprendizaje. En este sentido, es conveniente contar con algún mecanismo automatizado que permita detectar esos problemas para así poder ofrecer la ayuda en el momento adecuado y de la mejor manera. En esta situación, la tecnología de agentes inteligentes puede resultar beneficiosa, debido a que es capaz de evaluar las acciones de cada estudiante y detectar problemas, brindando la ayuda correspondiente. En este trabajo se presenta el desarrollo de un prototipo de sistema de recomendación de tutores, basado en una arquitectura multi-agentes para monitorear la interacción del estudiante con un entorno educativo virtual ubicuo en el nivel universitario, y detectar el tema en el que el estudiante presenta problemas. La recomendación de tutores se realiza a través de un mapa teniendo en cuenta sus ubicaciones, y de esa manera el estudiante tiene la libertad de acudir al más cercano. De las pruebas realizadas se demostró que el sistema propuesto facilita al estudiante la tarea de encontrar un tutor adecuado que se encuentre geográficamente cerca y lo pueda ayudar en el tema que presenta problemas.

Palabras clave: U-Learning; Sistemas multi-agentes; Recomendación de tutores.

Abstract

Devices such as mobile phones and tablets are allies of education, since they allow access to educational content and activities from anywhere and at any time. However, student support or assistance is not always available when a student has a learning problem. In this sense, it is convenient to have an automated mechanism that allows detecting these problems in order to offer them help at the right time and in the best way. In this situation, intelligent agent technology can be beneficial, because it is capable of evaluating the actions of each student and detecting problems, providing the corresponding help. In this work, development of a prototype of a tutor recommendation system is presented_. This system is_ based on a multiagent architecture and allows to monitor the interaction of the student with a ubiquitous virtual educational environment at the university level and to detect the learning subject that the student has problems with. The recommendation of tutors is made through a map taking into account their locations, enabling the student to attend to the closest one. The tests carried out show that the proposed system makes it easier for the student to find a suitable tutor who is geographically close and can help him in the issue that he has problems with.

Keywords: U-learning; Multi-agents systems; Recommendation systems.

1. Introducción

En los últimos años ha surgido un nuevo escenario educativo caracterizado por un conjunto de actividades formativas accesibles en cualquier lugar y desde cualquier dispositivo, el aprendizaje ubicuo (u-learning). El uso de dispositivos como, teléfonos móviles y tabletas, crean nuevas oportunidades para los estudiantes, quienes están intensamente conectados. De esta manera, se puede acceder a contenido educativo, y las interacciones se pueden llevar a cabo donde los estudiantes lo necesitan sin restricciones de espacio o tiempo [1].

Sin embargo, cuando un estudiante presenta algún problema en su aprendizaje en estos entornos, no siempre sabe a quién puede recurrir e incluso desconoce que la ayuda, en la mayoría de las veces, está más cerca de lo que cree.

En este sentido, un sistema de recomendación, que tenga en cuenta datos del contexto del estudiante, como la ubicación, lo puede asesorar en encontrar un tutor adecuado y que se encuentre físicamente cerca. En [2] se ha presentado un método para recomendar a los estudiantes expertos en el tema que quieren aprender, considerando para ello las experiencias de esos expertos con otros estudiantes, la disponibilidad de estos y la cercanía física con el estudiante para el caso de una consulta presencial.

En este trabajo se busca integrar dicho método, con una arquitectura multi-agente que permita detectar de manera automática los problemas que un estudiante presenta al resolver ciertos objetos de aprendizaje y recomiende aquellos tutores que conozcan del tema y se encuentren físicamente cerca.

Un sistema multi-agente es un consorcio de múltiples agentes trabajando juntos, donde cada agente se especializa en una función determinada [3]. Estos agentes son capaces de actuar de manera precisa con el fin de realizar tareas en representación de su usuario [4]. Para esto, se sitúan en un ambiente determinado y son capaces de ejecutar flexiblemente acciones autónomas con el fin de cumplir sus objetivos [5]. Esta flexibilidad se puede traducir como [6] reactividad o capacidad de percibir su ambiente y responder sin demoras a cambios que ocurren en él, pro-actividad o capacidad de exhibir un comportamiento dirigido a objetivos, tomando la iniciativa, y habilidad social o capacidad de interactuar con otros agentes (y posiblemente humanos) a través de un lenguaje de comunicación.

En la literatura existente, la integración de agentes en entornos de e-learning para facilitar la comunicación entre pares y con los docentes, por lo general, se produce a partir de un pedido de ayuda explícito del alumno y siempre teniendo en cuenta criterios de detección de problemas y de personalización de la ayuda, invariables. Es por eso que se propone construir una aplicación móvil que integrada con servicios web y sistemas multi-agentes, permita detectar las dificultades que se le presenten a un estudiante al resolver las actividades y evaluaciones propuestas en diferentes objetos de aprendizaje, y recomiende tutores que lo puedan ayudar en un entorno de aprendizaje ubicuo.

En la sección siguiente se presentan los trabajos relacionados encontrados en la literatura. Luego, en la sección 3, se describe la arquitectura multi-agente del sistema de recomendación. En la sección 4, se presenta el prototipo del sistema de recomendación propuesto. Finalmente, en la última sección, se expone la conclusión del trabajo.

2. Antecedentes

En esta sección se describen antecedentes relacionados a la recomendación de tutores empleando cualquier técnica para realizar la recomendación. A su vez, se describen trabajos que utilizan modelos multi-agentes para realizar la recomendación tanto de contenido como de tutores.

En [7] presentan un sistema multi-agente para u-learning sensible al contexto que ofrece planificación adaptativa de cursos virtuales, evaluación personalizada de cursos, selección de objetos de aprendizaje según el perfil del estudiante, búsqueda de objetos de aprendizaje en repositorios, búsqueda de asistentes de aprendizaje por temas y el acceso a diferentes actividades aprendizaje colaborativo sensibles al contexto. La arquitectura del sistema está compuesta por seis tipos de agentes: el usuario, el recomendador, el buscador, el planificador, el evaluador y el de contexto. La recomendación de asistentes por temas de aprendizaje ofrece a los estudiantes la posibilidad de buscar estudiantes avanzados que estén disponibles, que sean especialmente cercanos y que tengan conocimiento de las áreas de interés del estudiante.

En [8] presentan un sistema de recomendación de tutores pares basado en el aprendizaje automatizado con evaluación automatizada para mejorar el rendimiento de aprendizaje en la usabilidad de aplicaciones informáticas. El modelado para el aprendizaje automatizado está basado en red neuronal supervisada. Los estudiantes deben acceder a la aplicación web que contiene integrado los sistemas, de recomendación y de evaluación automatizado. En la aplicación, las tareas son asignadas a los estudiantes durante la clase y son corregidas automáticamente por el sistema de evaluación, permitiendo al estudiante tener una respuesta en tiempo real y determinar la necesidad de buscar la ayuda de tutores pares. La plataforma web ofrece la posibilidad de generar la recomendación de tutores cuando la solicitan, permitiendo generar el listado de tutores se ajusta a partir del rendimiento de aprendizaje, cuestionario completado, antes de comenzar la clase y de la retroalimentación de tutores que hayan brindado durante la ayuda.

En [9] presentan un sistema inteligente de recomendación de tutores para entornos de educación en línea (e-learning). La recomendación está basada en aprendizaje automatizado utilizando algoritmos de árbol de decisión. El entorno de prueba fue llevado a cabo en la plataforma

Tesys-Web donde habitualmente los estudiantes inician sesión para acceder a clases. Los mismos solicitan la ayuda de tutores y dan inicio al procedimiento de la recomendación. El árbol de decisión se construye a partir del historial de actividades y comunicación de cada estudiante que el sistema Tesys-Web almacena en la base de datos. El historial mencionado está conformado con los siguientes parámetros: identificador del estudiante, total de horas conectado en la plataforma, el promedio obtenido de una tarea, la longitud promedio de mensajes enviados y el promedio de la actividad de mensajería. Los nodos hojas representan perfiles de estudiantes para implementar la función de afinidad y complementar con el proceso de la recomendación.

En [10] presentan una plataforma de nivel de curso de código abierto llamada RiPPLE (Recomendación de Pares Personalizados en entornos de aprendizajes), ofrece a los estudiantes a contribuir en la creación de contenidos de aprendizaje, acceder a contenidos de aprendizajes adaptados a las necesidades del estudiantes y búsqueda de estudiante por temas. El algoritmo de recomendación recíproco tiene en cuenta la cantidad de estudiantes conectados, la cantidad de temas del curso, la cantidad de diferentes roles que abarca el estudiante durante la sesión). Los mismos, se combinan estratégicamente para generar la recomendación de estudiantes pares, la disponibilidad horaria y la preferencia del estudiante pueda elegir. La recomendación de pares ofrece a los estudiantes la posibilidad de buscar estudiantes avanzados que tengan disponibilidad horaria y que tengan conocimiento en el tema de interés seleccionado por parte del estudiante.

En [11] se presenta una interfaz adaptativa personalizada a un sistema multi-agente que recomienda OA (Objeto de Aprendizaje) a estudiantes, almacenados en repositorios heterogéneos incluyendo federaciones de repositorios, teniendo en cuenta las preferencias, características y necesidades de los estudiantes; lo anterior con el fin de mejorar la presentación, visualización y satisfacción de uso de dichos contenidos educativos. La arquitectura del sistema está compuesta por los siguientes agentes: agente interfaz: realiza la adaptación de la interfaz del repositorio, según las características que definen el perfil del estudiante que está realizando la búsqueda. Agente Usuario: gestiona el perfil del usuario, lo que implica crear y modificar tanto las características como sus preferencias. Agente Coordinador: redirecciona las consultas que hace el usuario al repositorio local y a los remotos para realizar la búsqueda. Además, accede a la información del agente usuario para conocer las características útiles para la recomendación. Agente repositorio local y agentes repositorios remotos: realizan las búsquedas propiamente en los repositorios, tanto de forma local como remota. Agente de recomendación: realiza el proceso de recomendación sobre los metadatos que describen los OA y en las informaciones de estilo de aprendizaje, grado de escolaridad y preferencia de idioma del estudiante registrado. Agente evaluador: encargado de gestionar las evaluaciones que realiza un usuario a algún OA que ha sido explorado.

En [12] se presenta un sistema que brinda sugerencias en tiempo real a usuarios, sobre diversos contenidos e información alojada en la web que esté relacionada con la temática estudiada en un momento dado dentro de un curso. Para la selección de los agentes a implementar en el prototipo de ayudas, fue necesario indagar en diferentes fuentes de información relacionadas con los agentes inteligentes y su comportamiento. Con la información obtenida, se concluyó que, para cada prototipo o desarrollo de un sistema basado en agentes, el autor elige y crea los agentes de acuerdo a los requerimientos del sistema, es decir, no hay especificaciones establecidas para nombrar y crear los agentes. Debido a esto, para el sistema de ayudas al usuario planteado se escogieron los agentes; Diagnóstico: detecta que el estudiante tarda en realizar las tareas, entonces informa a los demás agentes para iniciar la búsqueda de información. Móvil:viajan de un lugar de la red a otro para llevar a cabo tareas específicas de usuario. Notificación: avisan al usuario cuando ocurre un evento de interés para éste. Información: Filtran información electrónica basándose en las preferencias y gustos específicos de cada usuario. Recuperador: Proporcionan un conjunto de información personalizada al usuario de acuerdo con las preferencias de éste. Navegación: búsqueda de contenidos en la web.

A partir de los trabajos analizados, se puede observar que se implementan diferentes tecnologías para proporcionar una adecuada recomendación, ya sean de tutores o pares, contenidos e información, permitiendo satisfacer las necesidades del usuario. En cuanto a los trabajos que recomiendan tutores específicamente, la mayoría utiliza técnicas de aprendizaje automático basados en redes neuronales, árboles de decisión y no tienen en cuenta la posición de los estudiantes y tutores.

El único trabajo que emplea arquitectura multi-agentes y tiene en cuenta la posición del estudiante y tutor es [7], quienes recomiendan estudiantes avanzados que se encuentran cercanos al alumno en ese momento, pero parten de un pedido explícito de ayuda del estudiante.

3. Arquitectura Multi-Agente para la Recomendación de Tutores

Este trabajo propone un sistema de recomendación personalizada de tutores para u-learning, cómo herramienta de apoyo para los estudiantes que no saben a quién recurrir, o desconocen, que la ayuda está más cerca de lo que creen. El sistema tiene una arquitectura multi-agentes (Figura 1) basada en la arquitectura multi-agente propuesta en [13] y combinado con el método propuesto en [2] realiza la recomendación de tutores, una vez detectado que el estudiante presenta problemas en el aprendizaje de algún tema.

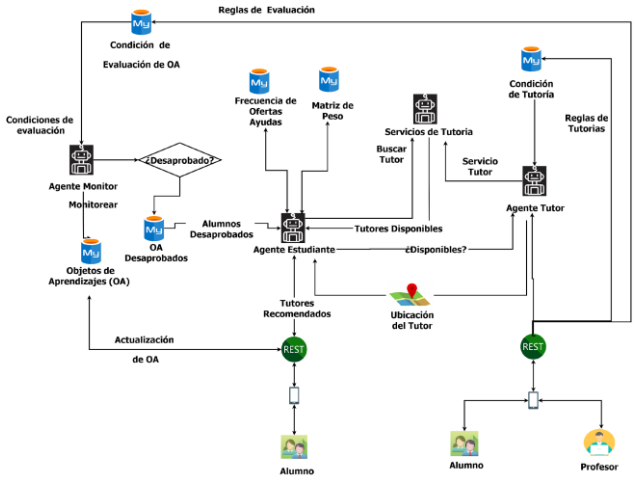

Figura 1. Arquitectura Multi-Agentes

Para la detección de problemas de aprendizaje, cada docente define los criterios por los cuáles considera que un estudiante presenta un problema de aprendizaje. Por ejemplo, se presentan diferentes objetos de aprendizaje (OA) a los estudiantes y se establece que, si un OA se encuentra desaprobado, el estudiante presenta problemas en ese tema en particular, por lo que se debe ofrecer la ayuda correspondiente. De esta manera, se definen diferentes condiciones de evaluaciones de OA que serán aplicadas correspondientemente.

A continuación, se detallará cada uno de los componentes del sistema de recomendación:

Agente Monitor: se encarga de monitorear y evaluar los resultados de los OAs. Este comportamiento, inicia de forma paralela diferentes máquinas de estados finitos, una por cada problema detectado. Estas máquinas constan de tres estados (Figura 2), *Notificando, Verificando y Finalizado*. El estado *Notificando*, registra en la base de datos (BD) el problema detectado en un determinado tema. Inmediatamente después, pasa al estado *Verificando* el que se encarga de consultar la BD de OAs desaprobados y se asegura de que el estudiante sea notificado de la detección del problema. Una vez que el estudiante recibe la notificación, se actualiza la BD para que no se vuelva a notificar al alumno por este mismo problema, pasando de esta manera al estado *Finalizado*.

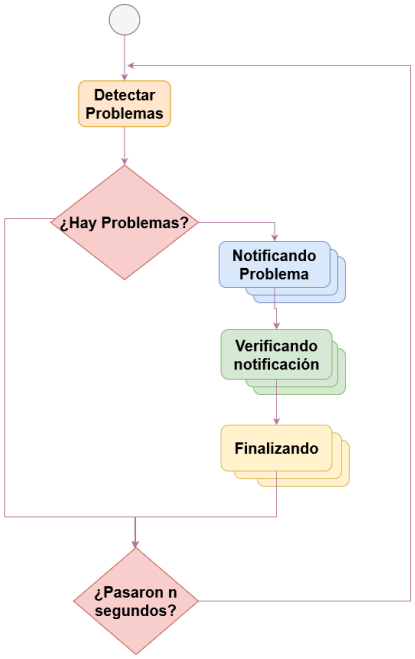

Figura 2. Funcionamiento del Agente Monitor

Agente Tutor: Se inicia cuando ingresan al sistema: un *Profesor* o un *alumno* que solo será tutor si cumple con las condiciones definidas en la BD de Condiciones de Tutoría que se describe más adelante. Automáticamente al iniciar, se suscribe a los Servicios de Tutoría informando el/ los tema/s en los que ofrece su servicio. Este agente está compuesto por tres comportamientos *RecibirPedidos, AtenderConsultas y LiberarCupos*, los que se ejecutan repetidamente hasta que la ejecución del agente finaliza. El comportamiento *RecibirPedidos* está a la espera de la solicitud de tutoría por parte de un Agente Estudiante. A este pedido se contesta en función del cupo de estudiantes que puede atender un tutor de forma simultánea. Se acepta el pedido si el tutor tiene cupo disponible y le envía la ubicación del mismo. Y no se acepta el pedido en caso de que el tutor esté atendiendo el máximo de alumnos posibles. Este cupo es actualizado por los comportamientos *AtenderConsulta y LiberarCupo*. El comportamiento *AtenderConsulta* espera un mensaje del agente Estudiante, donde le informa al Tutor que desea iniciar una consulta sobre un determinado tema. De esta forma, se decrementa el valor del cupo para ese Tutor. Por otra parte, el comportamiento *LiberarCupo* espera un mensaje con la notificación de que la consulta ha finalizado, e incrementa el cupo.

Agente Estudiante: Se inicia cuando un estudiante ingresa en el sistema. Está compuesto por una máquina finita de trece estados (Figura 3).

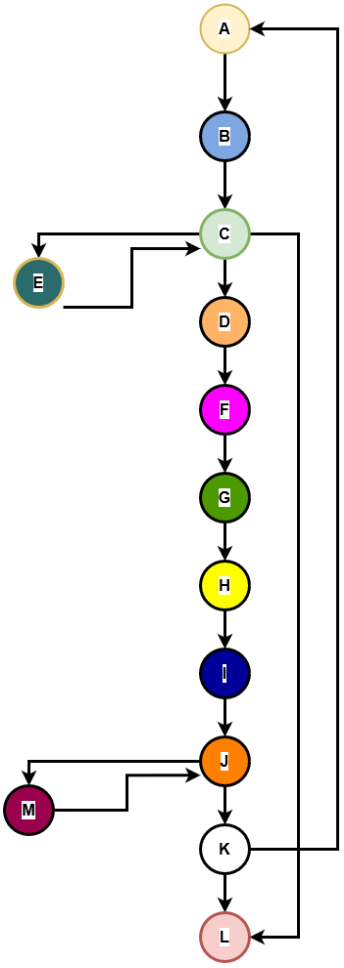

Figura 3. Funcionamiento del Agente Estudiante

Los estados se describen a continuación:

El estado *EsperandoEvaluación (Estado A):* consulta en la BD de OA desaprobados para verificar si el estudiante presenta un problema en algún tema.

El estado *NotificarEstudiante (Estado B):* informa al estudiante el problema y ofrece ayuda.

El estado *EsperandoAceptación (Estado C):* espera la respuesta del estudiante, en el caso que acepte, pasa al estado de *BuscandoTutores (Estado D)*. En caso de finalizar, pasa al estado de *Fin (Estado L)* y en el caso de esperar, significa que el estudiante no ha tomado ninguna decisión entonces pasa al estado *EsperarDecisión (Estado E).*

El estado *EsperarDecisión (Estado E)*, deja pasar unos minutos, dando la posibilidad de que el estudiante confirme su decisión. En caso de no recibir respuesta, vuelve a consultar a través del estado *EsperandoAceptación (Estado C).*

Una vez que se obtiene una lista de tutores en el tema en cuestión, el estado *EnviandoSolicitud (Estado F)*, envía un mensaje a cada uno de los tutores de la lista, consultando la disponibilidad. En el estado *EsperandoRespuestas (Estado G)*, se recepcionan las respuestas de los agentes tutores informando la disponibilidad del tutor.

En caso de que ningún tutor se encuentre disponible, pasa al estado *Espera (Estado H)*, donde se deja pasar unos minutos, dando la posibilidad de que un tutor se desocupe. Luego, se vuelve a consultar a través del estado *EnviandoSolicitud.*

Al recibir al menos un tutor disponible, pasa al estado *ListarRecomendacion (Estado I)*, el que implementa el método propuesto en [2] para generar la lista de tutores y así generar la recomendación. Para hacerlo, tiene en cuenta *la Matriz de Pesos* y *Frecuencia de Oferta de Ayuda*, que se describirán más adelante. Esta recomendación es enviada al estudiante (Figura 4). Una vez elegido el tutor se visualizara la ubicación, tal como se puede ver en la Figura 5.

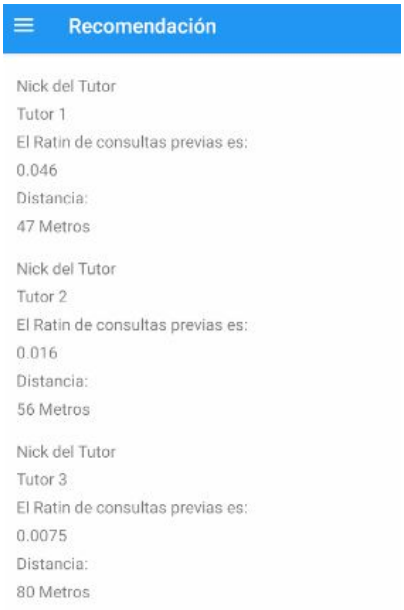

Figura 4. Listado de Recomendación de Tutores

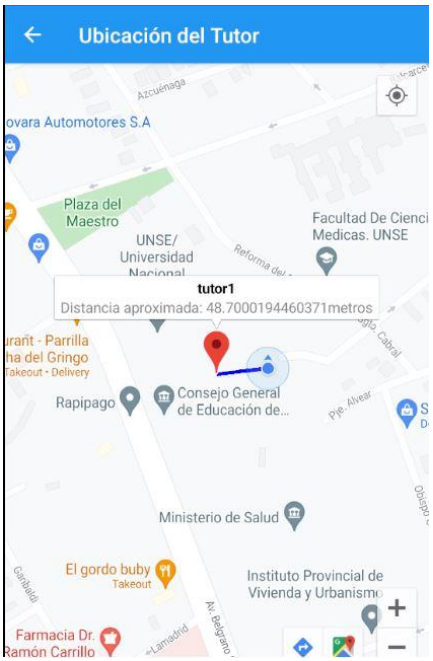

Figura 5. Ubicación del Tutor seleccionado

El estado *NoficandoFinalización (Estado J)*, verifica que el estudiante haya finalizado la consulta y la registra en la base de datos. Si la consulta finalizó, envía al Agente Tutor un mensaje del tipo INFORM notificando esta situación, desuscribe al Agente Estudiante del DF, para que no siga recibiendo información sobre nuevos tutores que ofrecen servicios en el tema y pasa al estado *Finalizado (Estado K)*. En caso contrario, pasa al estado *EsperarCerrarConsulta (Estado M)*.

En el estado *EsperarCerrarConsulta (Estado M)*, se deja pasar unos minutos dándole la posibilidad que el estudiante finalice la consulta. Luego, se vuelve a consultar a través del estado *NoficandoFinalización.*

En el estado *Finalizado (Estado K)*, se consulta al alumno si desea continuar recibiendo este tipo de ayuda. En caso de obtener respuesta afirmativa, vuelve al estado inicial *EsperandoEvaluacion*. Si no desea recibir más ayuda, pasa al estado final *Fin (Estado L).*

Servicio de Tutoría: Registra los servicios de tutoría asociado a cada tema que un agente tutor puede brindar. Permite a cada agente estudiante consultar los servicios que requieran para iniciar con el proceso de recomendación.

Matriz de Pesos (MP): Representa las preferencias de cada $estudiante\ i$ que haya aceptado la ayuda y es expresada como un vector de pesos $[w_{i1}, w_{i2}, \ldots, w_{in}]$, donde cada w_{ij} es calculada de acuerdo a la *fórmula (1)*:

$$w_{ij} = \frac{X}{Y} \qquad (1)$$

donde ***X*** es el número de veces que el experto *j* asesoró al alumno *i* y **Y** es el número total de veces que el alumno *i* fue asesorado por expertos en ese tema. Esta información se obtiene de la Base de datos (BD) frecuencias de ofertas de ayudas. Si se consideran los vectores ponderados de todos los estudiantes que forman el **conjunto E**, es posible construir la **Tabla 1:**

Tabla 1. Matriz de Peso

Estudiantes / Tutores	T_1	T_2	...	T_m
E_1	W_{11}	W_{12}	...	W_{1m}
E_2	W_{21}	W_{22}	...	W_{2m}
...	…	…	...	…
E_n	W_{n1}	W_{n2}	...	W_{nm}

Cálculo de la función de puntuación para la recomendación: esta función permite obtener un vector utilizando los valores ponderados de cada tutor sobre el tema, obtenido de **MP**. Se calcula con la siguiente *fórmula (2)*:

$$TP_j = (w_{ij})\frac{1}{d_j};\ j = 1 \ldots n \qquad (2)$$

Entonces, los elementos del vector se clasifican desde arriba a abajo y la recomendación de tutores para el estudiante es generada.

Base de datos de Condición de Evaluación de OA: contiene las condiciones de evaluación y OAs, configurables por los profesores. Cada condición está asociada a los Objetos de Aprendizajes definidos por el mismo profesor, para determinar cuándo se considera que un estudiante presenta problemas en su correspondiente evaluación.

Además, será consultado por el agente Monitor, para obtener las reglas de evaluación para evaluar los resultados de los OA. Las posibles reglas son las siguientes:

- El resultado obtenido por el estudiante debe ser menor o igual al valor promedio definido por el profesor.
- El estudiante desaprueba la evaluación del OA.

Base de datos de Condiciones de Tutoría: esta base de datos define las políticas que determinan cuando un estudiante puede ser tutor. De esta manera, cuando un estudiante ingresa al sistema, se verifica si cumple con las condiciones de tutoría para iniciar el Agente Tutor. Las posibles reglas de tutoría son las siguientes:

- El resultado obtenido en un OA supera o es igual a un valor promedio definido por el profesor.
- El estudiante aprobó el OA.

Base de datos de OAs: contiene características de los OA que simulan cursos, los datos contenidos son: nombre de la materia al que pertenece, descripción del tema, identificador del profesor que la dicta, estado que representa el resultado y toma el valor aprobado o desaprobado y nota que representa el valor numérico que obtuvo al resolver el OA. El agente Monitor se encarga de monitorear en función de las reglas extraídas del almacén **Condición de Evaluación OA**, para almacenar en **Registro de OA Desaprobados.**

Base de datos de Registro de OA Desaprobados: contiene datos de los estudiantes que desaprobaron algún Objetos de Aprendizaje.

Base de datos de Frecuencia Ofertas Ayudas: contiene registros de las consultas atendidas por cada tutor. Cada consulta está formada por el identificador del tutor, el nivel de servicio asignado por el estudiante y la descripción del tema del objeto de aprendizaje. El agente estudiante recopila el nivel de servicio del tutor disponible para brindar la ayuda que sea solicitada.

4. Prototipo de la Arquitectura Multi-Agente para la Recomendación de Tutores

Para el desarrollo del prototipo se trabajó con una arquitectura basada en servicios Web, la que implementa el funcionamiento de los agentes. Se utilizó el lenguaje Java, la librería JADE (JAVA Agent Development Framework) [14], base de datos MySQL y la librería Jersey-ws de lenguaje Java, para el desarrollo de Apis Web Service Full Rest, permitiendo de esta forma la interacción entre los usuarios y agentes.

Los servicios de tutoría se implementaron a través del Agente DF de JADE, componente que provee el servicio de páginas amarillas a otros agentes, donde éstos pueden buscar o publicar diferentes servicios [12].

Los usuarios, acceden a estos servicios mediante una aplicación móvil, la que fue desarrollada en lenguaje C# y XAML utilizando la api de Xamarin.Forms, generando una aplicación nativa en sistemas operativos Android y IOS.

La arquitectura mencionada, fue implementada en un entorno simulado, donde se adoptaron las siguientes políticas de sistema que podrían ser modificadas por otras implementaciones:

- **Política de detección de problemas:** el resultado puede ser calificado como desaprobado o aprobado.
- **Política de tutoría estudiantil:** un alumno puede ofrecer servicios de tutoría en un tema si ha aprobado el OA correspondiente al mismo.
- **Política de frecuencia de ofrecimiento de ayuda:** se volverá a ofrecer ayuda al estudiante tanto en el caso de que el alumno acepte la ayuda y no concrete la comunicación con algún tutor.
- **Política de asignación de tutores:** los tutores se asignan de forma balanceada no excediendo los tres alumnos por tutor.

Para ejecutar las pruebas de simulación, se trabajó con una cantidad de 10 estudiantes y 3 tutores, de los cuales: 4 estudiantes desaprobaron OAs del tema de Fundamentos de la Programación, 3 estudiantes desaprobaron OAs del tema Simulación y 3 estudiantes desaprobaron OAs del tema Redes. Además, se ha utilizando una herramienta de software de generación de datos de prueba, para simular las diferentes ubicaciones de los tutores y estudiantes. La herramienta fue realizada en el marco del proyecto de investigación “Métodos y Técnicas para desarrollo de Aplicaciones Ubicuas" de la Universidad Nacional de Santiago del Estero.

A continuación, se describe un escenario concreto extraído de las pruebas realizadas, a modo de ejemplo del funcionamiento de la aplicación con las políticas configuradas:

El **agente monitor** monitorea constantemente los OAs almacenados en la BD de OAs. En este escenario, se evaluarán sólo los resultados de Desaprobados o Aprobados para registrar en la **BD de OA Desaprobados**, de acuerdo a las políticas definidas en la **BD Condición de Evaluación**.

El alumno Juan accede al sistema, ingresando las credenciales de usuario y contraseña en el login de la aplicación móvil. Valida las credenciales y se inicia el Agente Estudiante para notificar los resultados de los OAs. Desde la aplicación móvil, Juan selecciona historial de evaluaciones del menú de opciones, para visualizar los resultados de los OA realizados (Figura 6).

Figura 6. Historial de Evaluaciones

El Agente Estudiante, al detectar que existen OA desaprobados, ofrece ayuda (Figura 7).

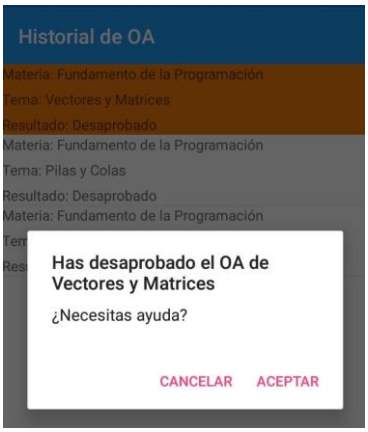

Figura 7. Oferta de Ayuda

Juan acepta la ayuda y se solicita al **Servicio de Tutoría**, los datos de tutores que puedan ayudarlo en el tema del OA desaprobado. En el caso que haya transcurrido 1 minuto y el **agente estudiante** no haya recibido una respuesta por parte del alumno, preguntará nuevamente (Figura 6). Pero, si Juan presiona en *Cancelar* ante el ofrecimiento de ayuda, entonces la recomendación se cancela**.** En ese momento se encuentran conectados tres expertos en el tema con sus respectivos **Agentes Tutores** iniciados, Tutor 1, 2 y 3. Cada **Agente Tutor** verifica el cupo disponible para las consultas. Si el cupo es menor al límite establecido, entonces envía la ubicación del tutor correspondiente. Caso contrario, el **Agente Estudiante** de Juan espera que se habilite un cupo o acceda al sistema, en ese momento, otro tutor para ese tema. Luego de haber

recibido la respuesta de los **agentes tutores**, se calcula la distancia entre el estudiante y los tutores, como se puede visualizar en la **Tabla 2**.

Tabla 2. Distancia de los tutores hacia al estudiante

Tutores	**Distancia hacia el estudiante**
Tutor 1	**47 mts**
Tutor 2	**56 mts**
Tutor 3	**80 mts**

A su vez, existen cuatro estudiantes que previamente consultaron sobre el tema de Fundamentos de la Programación: Pablo, Milagros, Maximiliano y Luján. La cantidad de veces que consultaron cada estudiante a cada tutor se visualiza en la **Tabla 3**.

Tabla 3. Número de veces que los estudiantes consultaron a los tutores por el tema

	Tutor 1	**Tutor 2**	**Tutor3**	**Total**
Pablo	**2**	**2**	**1**	**5**
Milagros	**3**	**2**	**3**	**8**
Maximliano	**1**			
Luján	**2**	**1**		**3**

Se puede observar en la **Tabla 3**, el total de veces que los estudiantes consultaron el tema *Vectores y Matrices* y la cantidad de veces que cada estudiante consultó a cada tutor. En este caso, el estudiante Pablo, realizó un total de 5 consultas, representando el parámetro "Y", y solo 2 de estas consultas fueron realizadas al Tutor 1, lo que representa el parámetro "X". Estos datos mencionados serán extraídos de la **BD Frecuencias de Oferta de Ayuda**, para que el **Agente Estudiante** aplique la **Fórmula 1** y obtener el peso (w): 0.4. Se aplica dicha fórmula a todos los estudiantes/tutores de la **Tabla 3** y se obtiene la **Matriz de Pesos (Tabla 4**). Luego, se aplica la **Fórmula 2** por el Agente Estudiante, se obtiene la suma de pesos de cada tutor. Se puede observar que el tutor más consultado es el Tutor 1 y el menos consultado es Tutor 3.

Tabla 4. Matriz de Peso con Totales

	Tutor 1	**Tutor 2**	**Tutor 3**
Pablo	**0.4**	**0.4**	**0.2**
Milagros	**0.3**	**0.2**	**0.3**
Maximiliano	**1**		
Luján	**0.6**	**0.3**	
Total	**2.3**	**0.9**	**0.3**

Los elementos del vector de distancia (Tabla 2) son usados para generar una ponderación (1/d), que es multiplicada por el total de pesos correspondiente al tutor, que fue obtenida en la matriz de pesos (Tabla 4). Entonces, para el Tutor 1 será:

2.3 x 0.02 = 0.046

Se repite el procedimiento para todos los tutores, y se obtiene el siguiente **vector ET**, lo que permite ordenar la lista de tutores a recomendar:

ET = <0.046,0.016, 0.0075 >

Finalmente, el **Agente Estudiante** de Juan informa la recomendación con las correspondientes distancias aproximadas de los tutores y el rating de servicio, cómo se puede ver en la Figura 8.

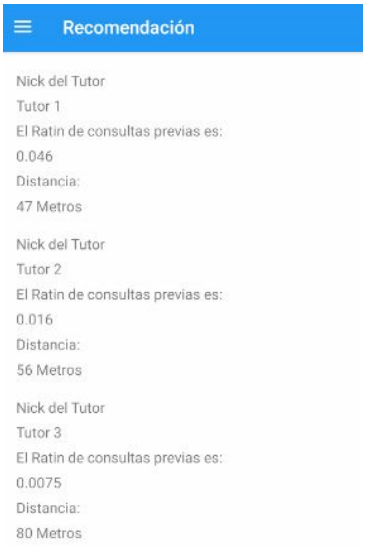

Figura 8. Listado de Recomendación de Tutores

Para conocer la posición física del tutor, Juan debe seleccionar del listado recomendado un tutor, ya sea que le parezca más cercano o el que mayor rating tenga. Entonces una vez seleccionado, la aplicación mostrará y guiara, a través del mapa, hacia la ubicación del tutor (Figura 9). Cuando Juan llega hasta el destino, se notifica (Figura 10).

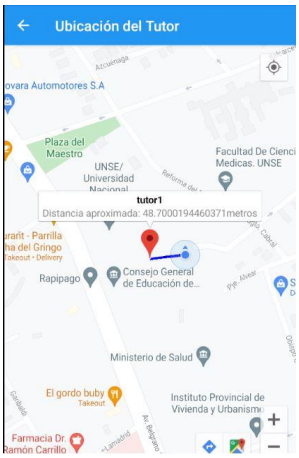

Figura 9. Tutor Seleccionado

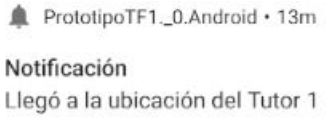

Figura 10. Notificación de proximidad

Desde el momento que se detecta que tanto el estudiante como el tutor se encuentran en el mismo lugar, el **Agente Tutor** decrementa el cupo del tutor e inicia el cronómetro para controlar el tiempo de duración de la consulta, como se puede ver en la (Figura 11).

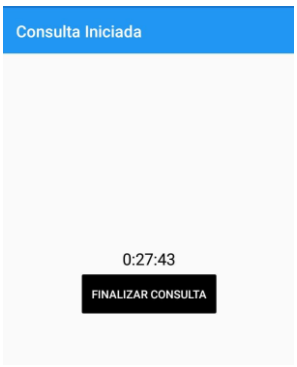

Figura 11. Consulta Finalizada

Una vez que Juan presiona el botón de “Finalizar Consulta”, el **Agente Tutor** incrementa el cupo disponible habilitando un lugar más para los estudiantes que quieran realizar una nueva consulta. Por otro lado, el tutor y alumno serán datos que se tendrán en cuenta para actualizar la ***Tabla 3*** para futuras recomendaciones.

Conclusiones

En este trabajo se presentó un sistema de recomendación de tutores para entornos de aprendizaje ubicuo basado en una arquitectura multi-agentes, que tiene en cuenta la ubicación del estudiante y de los tutores para realizar la recomendación. Se demostró que la arquitectura multiagentes proporciona un medio óptimo y eficaz para la implementación de sistemas como el presentado en este trabajo, debido a la capacidad autónoma, reactiva, proactiva que poseen los agentes. Estas capacidades de los agentes permiten el monitoreo constante de las actividades de los estudiantes y facilitan la labor de encontrar la mejor opción de tutoría en el momento justo que el alumno lo requiere.

La posibilidad de definir políticas parametrizables por cada docente en la detección de problemas de aprendizaje, como al momento de determinar los estudiantes que pueden ser tutores en determinados temas, le da versatilidad a la aplicación y le provee de más herramientas a los docentes que desean acompañar el aprendizaje de sus alumnos.

Debido a la situación de pandemia vivida durante este año, no se pudieron realizar las pruebas del sistema en un contexto real, pero es el trabajo a futuro que se piensa abordar.

Es importante destacar que, sobre la base del sistema presentado, se podrían agregar características adicionales como: la calificación otorgada por el estudiante tras haber finalizado la consulta, el orden de tutores seleccionados y las cantidades de veces que el alumno elige al tutor para realizar consultas. Con las características mencionadas se espera mejorar las recomendaciones a los estudiantes, así como también otras formas de comunicación entre los estudiantes y los tutores.

Referencias

[1] S. Graf and Kinshuk, “Adaptivity and personalization in ubiquitous learning systems” in *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2008, pp. 331-338.

[2] E. B. Duran and M. M. Alvarez, “Method for generating expert recommendations to advise students on ubiquitous learning experiences,” in *Proceedings - International Conference of the Chilean Computer Science Society*, 2017, pp. 1-8.

[3] P. S. Sajja and R. Akerkar, *Intelligent Technologies for Web Applications*. 1st ed. Chapman & Hall/CRC, 2012.

[4] H. S. Nwana, “Software agents: An overview,” *Knowledge Engineering Review*, 1996, pp. 205-244.

[5] M. Wooldridge, *Introduction to MultiAgent Systems*. Wiley & Sons, 2002.

[6] M. Wooldridge and N. R. Jennings, “Intelligent agents: Theory and practice,” *The Knowledge Engineering*, pp.115– 152, 1995.

[7] D. A. Ovalle, O. M. Salazar and N. D. Duque, “Customized recommendation model for virtual courses based on ubiquitous computing and intelligent agents | Modelo de recomendación personalizada en cursos virtuales basado en computación ubicua y agentes inteligentes,” *Inf. Tecnol.*, vol. 25, no. 6, 2014.

[8] Z. H. Ma, W. Y. Hwang and T. K. Shih, “Effects of a peer tutor recommender system (PTRS) with machine learning and automated assessment on vocational high school students’ computer application operating skills,” *Journal of Computers in Education*, vol. 7, no. 3, pp 435–462, 2020.

[9] M . C. Mihaescu and P. S. Popescu and C. Ionascu, “Intelligent Tutor Recommender System for On-Line Educational Environments,” in *Proceedings of the 8th International Conference on Educational Data Mining*, 2015, pp. 516–519.

[10] B. A. Potts, H. Khosravi, C. Reidsema, A. Bakharia, M. Belonogoff and M. Fleming, “Reciprocal peer recommendation for learning purposes,” in *ACM International Conference Proceeding Series*, 2018.

[11] O. M. Salazar Ospina, P. A. Rodríguez Marín, D. A. Ovalle Carranza and N. D. Duque Méndez, “Interfaces

adaptativas personalizadas para brindar recomendaciones en repositorios de objetos de aprendizaje”, *Tecnura*, vol. 21, no. 53, pp. 107–118, jul. 2017.

[12] C. E. Valencia Carabali and E. M. Ordoñez Palechor, “Desarrollo De un Sistema de Ayuda al Usuario Basado En Agentes Inteligentes Para La Plataforma Informática Educativa UCC” Tesina de Grado, Facultad De Ingenieria, Universidad Cooperativa de Colombia, 2018. [Online]. Available: https://repository.ucc.edu.co/bitstream/20.500.12494/8523/1/2018_desarrollo_sistema_ayuda.pdf

[13] B. F. Reuter, M. M. Alvarez, G. Gonzalez, and E. B. Durán, “Multi-agent system model for tutor recommendation in ubiquitous learning environments,” *Workshop on Advanced Virtual Environments and Education*, vol. 1, no. 1, pp. 10-17, 2018.

[14] F. Bellifemine, G. Caire and D. Greenwood, *Developing Multi-Agent Systems with JADE*. 2007. [Online]. Available: https://doi.org/10.1002/9780470058411

Información de Contacto de los/as Autores/as:

Luciano Juárez
Quintana N° 429, La Banda
Santiago Del Estero
Argentina
lucianojuarezgaston@gmail.com
ORCID iD: https://orcid.org/0000-0003-2574-9619

Beatriz Reuter – Fernández
24 de Septiembre 520
Santiago Del Estero
Argentina
bfreuter@unse.edu.ar
ORCID iD: https://orcid.org/0000-0002-2134-9821

Elena Durán
Islas Malvinas N°785, Barrio Sáenz Peña
Santiago Del Estero
Argentina
eduran@unse.edu.ar
ORCID iD: https://orcid.org/0000-0003-3978-5194